Text to vector

30 June 2021 15:28

Text to vector

BoW :-

Bag of Words :

It is a technique to convert a text to vector.

R1: This pasta is tasty and delicious

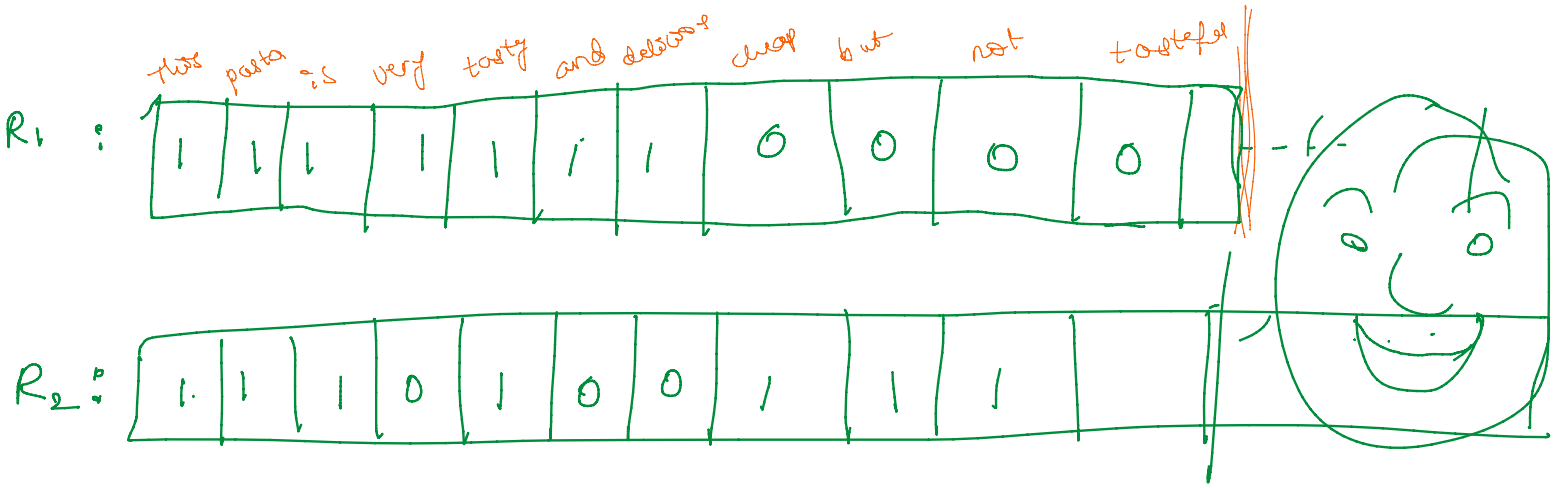
R2: This pasta is not tasty but affordable

R3: This pasta is delicious and cheap

R4: Pasta is tasteful.

Now first we need to form vocabulary of all unique words.

{This, pasta, is, tasty, and, delicious, affordable, but, not, cheap, tasteful.}



So we fill the counts of the words in our text to form vector.

Here we consider #counts.

form vector.

Here we consider #counts.

For binary BoW, { 1 if w_i present
0 if w_i not present.

* Here we can see that R_1 & R_2 are opposite, but distance bet^n them is low.

* Also BoW doesn't take consideration of semantic meaning

Unigram – Bigram – trigram —— n-gram

Previous example is unigram.

If we consider pair of words to construct vector, that is called bigram.

If we consider n-words at a time to construct vector
——→ n-gram

Text Preprocessing steps:-

1). Stop word Removal

"Not always a better choice"

2) Convert to lower case

3) Remove special characters

4) Short form → full form

5) Stemming
   * Snowball stemmer → New.
   * Porter Stemer → old

Converts. study, studying, studies → studi

No grammer rules

6). Lemmatization

Considerations of Parts of Speech:

go went gone → go

Praveen Hegde
CONFIDENTIAL

Lemmatization usually refers to **doing** things properly with the use of a vocabulary and morphological analysis of **words**, normally aiming to remove inflectional endings only and to return the base or dictionary form of a **word**, which **is** known as the **lemma** .

It means understanding the structure of text, **Morphological analysis** is the process of providing grammatical information about the word on the basis of properties of the morpheme(basic unit that can not be further divided) it contains.

In order to create the BoW firstly, we are required to remove the stop words. Then we process our Stemming process. Stemming is the algorithmic process where each words having the same sense of understanding are converted to its stem or root. Here root can be a word in itself or not. Example:
study , studying , studies all stemmed to studi (not study). Each word can be formed by adding the suffixes to the root.
We also perform Lemmatisation which is the algorithmic process of combining the inflected words into a common word called lemma. The Algorithm make use of the

dictionary to guess down the lemma . Suppose we have words = {go , going , went} we get lemma as "go".
If you want to know how nltk does it let me give u a rough idea of it.

NLTK makes use of Parts of speech - Tagging , where parts of speech tagger take group of words as an input and it returns list of tuples. Tuple contains a word and corresponding parts of speech.This parts of speech tagging is done to make sense of context in the sentence and helps lemmatizer to choose the appropriate lemma.

.

Tokenization :—

Tokenization is basically breaking our paragraphs into list of sentences and sentences into list of words

New York ⟶ New_York

New Delhi ⟶ New_Delhi

After these preprocessing, we apply BoW / any vectorization technique.

though we can't preserve semantic meaning

TF-IDF :⟶

Term frequency Inverse-Document frequency :—

Term frequency,

$$TF(w_i, r_j) = \frac{\#\text{ times } w_i \text{ occurs in } r_j}{\text{total } \#\text{ words in } r_j}$$

$0 \leq TF(w_i, r_j) \leq 1$

$w_i \rightarrow$ ith word

$r_j \rightarrow j^{th}$ document.

i.e. w_i is the i^{th} word in j^{th} document.

i.e. TF is frequency of word in particular document.

IDF :→

$$IDF(w_i, D_c) = \log\left(\frac{N}{n_i}\right)$$

$N \rightarrow$ total # documents

$D_c \rightarrow$ text corpus.

$n_i \rightarrow$ # documents in which w_i occurs.

So IDF is calculated for word w_i in consideration with total text corpus D_c.

always, $N \geq n_i$

$$\Rightarrow \frac{N}{n_i} \geq 1 \Rightarrow \log\left(\frac{N}{n_i}\right) \geq 0$$

$$\Rightarrow IDF(w_i, D_c) \geq 0$$

As $n_i \uparrow^{seq} \rightarrow \frac{N}{n_i} \downarrow_{seq} \rightarrow \log\left(\frac{N}{n_i}\right) \downarrow_{seq}$

$$\Rightarrow IDF(w_i, D_c) \downarrow_{seq}.$$

i.e.

IDF value will be low for most repeating words in D_c

As w_i repeat more in D_c, $IDF(w_i, D_c) \downarrow_{seq}$

If n_i ↓ $\rightarrow \left(\frac{N}{n_i}\right)$ ↑ $\rightarrow \log\left(\frac{N}{n_i}\right)$ ↑

IDF(w_i, D) ↑

i.e. IDF value will be high for rare word in corpus.

- tfidf$(w_i, r_j) \rightarrow$ TF(w_i, r_j) * IDF(w_i, D_c)

TF ↑ → most frequent word in particular document.

IDF ↑ → Rare word in total text corpus.

Why log in idf :—

there is no strong theory behind this.

Some intution;

Let $N = 1000$

	$\frac{N}{n_i}$	$\log\left(\frac{N}{n_i}\right)$
The (will present almost all documents)	$\frac{1000}{1000} = 1$	0
is	$\frac{1000}{1000} = 1$	0
⋮	⋮	
civilization (very rare)	$\frac{1000}{1} = 1000$	≈ 6.9

large gap

gap is very small

civilization ...

Hence if we don't consider log, $\frac{N}{n_i}$ value will dominate in tf * idf

log will scale the values to smaller values

W2V :-

W2V is state of the art technique.

It considers semantic meaning.

Its maths is very rigourous and discussed in Deep Learning

$V_i \rightarrow$ Vector representation of W_i

Google has trained using google news data

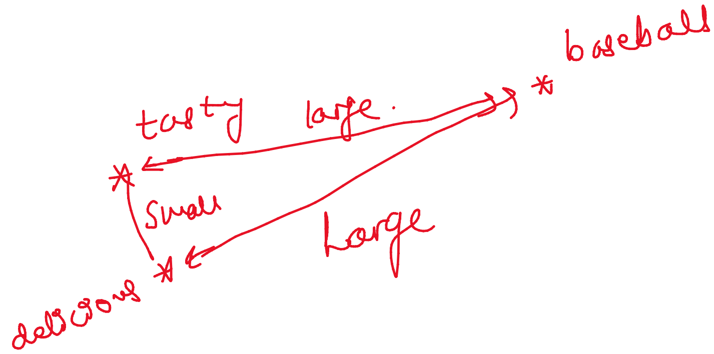
Its 300-d vectors for each word.

(Glove model)

(GloVe model)

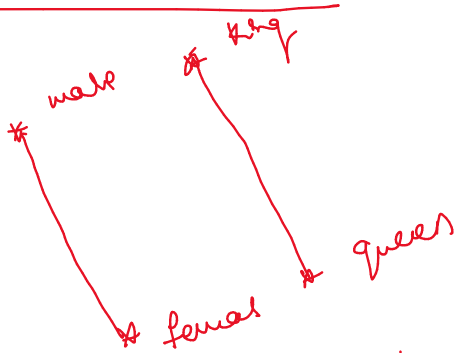
What W2V does is,

1) It preserves semantical meaning.



If, words are semantically similar } their vectors are closer

2) Identifies Relationships.



$$(V_m - V_f) \parallel (V_k - V_q)$$

almost like Eq

IIIly for country - capitals also.

---

larger the dimension of vector

↳ more information.

Avg W2V

$R_1 \rightarrow w_1\ w_2\ w_3\ w_4$

$w_1 \rightarrow$  V_1

$w_2 \rightarrow$  V_2

w_3 & w_4.

$$\text{Avg } W2V \rightarrow \frac{\sum_{i=1}^{n} V_i}{N}$$

tf-idf W2V :-

$$= \frac{t_1 * w2v(w_1) + t_2 * w2v(w_2) + t_3 * w2v(w_3) + \cdots + t_n * w2v(w_n)}{\sum_{i=1}^{n} t_i}$$

$t_i \rightarrow$ tfidf value of i^{th} word.

Avg W2V & tfidf W2V are weighing schemes.